Editor: Donald J. Reifer ■ Reifer Consultants ■ d.reifer@ieee.org

Seven Pitfalls To Avoid in the Hunt for Best Practices

Richard Turner

A recent US Defense Science Board study recommended the identification and implementation of best practices as a critical need for successful acquisition programs. Practices in this sense go beyond the development-related guidance of the popular Capability Maturity Models, including approaches organizations use to manage others creating software for them under contract. The study identified several traditional project-management practices such as configuration management and risk management, as well as newer practices such as executable architectures and limiting development time.

Little real implementation

In my role to support the transition of new software-intensive system-acquisition technologies to the US Department of Defense workforce, I looked into the adoption of best practices in defense acquisitions. My research uncovered considerable recognition of the most widely referenced best practices, but very little real implementation. As I suspected, there were good reasons for the unsuccessful implementation of even the highly recommended practices. I now believe these reasons apply to the application of practices in nearly any situation—commercial or government, acquisition or development.

I found that practices—best or otherwise—generally don't fit into all contexts, and it isn't easy to evaluate how appropriate a practice is for a particular organization or program. While there might be a few universally good practices (such as peer reviews), and there are certainly required practices (such as configuration and requirements management), most practices have hidden assumptions and conditions for use, and there is little empirical data to support evaluation and selection. Consequently, managers often find themselves on a veritable treasure hunt for practices that will actually help their efforts rather than prove to be only fool's gold.

Adopting this treasure-seeker metaphor, even the most experienced adventurer must identify the dangers and risks that might be faced. I've scouted out seven pitfalls to watch for in assessing best practices, and suggest some questions to ask to help you avoid them. While most of these pitfalls aren't "deadly," they can certainly raise your costs, lengthen your schedule, and increase your risk—as well as possibly damage your professional reputation. There is also synergy and overlap between the pitfalls; one can lead to or camouflage another.

Common hazards

Let's consider the most frequently encountered hazards on the best practice treasure hunt.

Insufficient supplies: How much will it really cost?

Probably the most important question to ask of any practice is the size of the bill. This is a difficult question to answer, but you

0740-7459/03/$19.00 © 2003 IEEE

should at least consider the major costs. How many hours will you spend training your staff? What type of additional tools might be required? The initial costs might just be the tip of the iceberg. What is the cost in effort and resources of actually applying the practice? How much infrastructure will require funding to support? Are there license fees or equipment maintenance associated with it? Know the costs early to avoid nasty surprises.

Arcane treasure: What's the actual benefit?

The second pitfall follows on the heels of the first. What exactly will the program get from implementing the practice? Will it shorten the schedule, raise quality, or lower cost or risk? If so, by how much? How are benefits measured? Sometimes there are hidden benefits, or ones that surface late in the life cycle. On the other hand, also identify any hidden requirements for achieving stated benefits. Even obvious benefits might need actions outside your software development practices to be fully realized. For example, peer programming can provide higher quality and shorter development times, but successful implementation might require changes to corporate policy regarding reward structure, office space, or equipment allocations. Dig deep to make sure you uncover everything you can so there won't be disappointments down the road.

False leads: What's the pedigree?

There is no Underwriter's Laboratory or Good Housekeeping seal to certify practices, so identifying the source can be important. Who actually established this "best practice"? Is it technologically mature or the latest idea from a popular magazine? Are there empirical studies that suggest it works? Do you know anyone who has successfully implemented it? This is especially true when proprietary components such as tools or processes are part of the practice. Caveat emptor is a very good rule of thumb to follow.

The wrong map: What environment is assumed?

Every practice doesn't apply to every type of project. Does the practice assume a particular type of development environment? Does it only work for small projects? Was the best practice identified in an environment of stable requirements or is its primary benefit realized only in a situation of constant change? When in the product life cycle is it best applied? Implementing a requirements practice might not be helpful when you are knee deep in design. What is the size or criticality threshold at which the practice begins to pay off? What is reasonable for the F-22 fighter jet or a ballistic missile defense system might not be appropriate for your Web-enabled business application.

Your project team's attitude and personality are among the biggest things that affect the use of practices. Will they accept and adopt the practice or just go through the motions?

Missing mileposts: How long until it works?

The time it takes for a benefit to be realized is one of the subtlest pitfalls. Does the practice provide immediate benefit, or do the effects have to trickle down through your development or acquisition process for several months (or years) before it actually bears fruit? Compare the benefit latency of peer reviews with that of a process improvement program. The first pays dividends immediately while the second might not have an impact for months or years. Understanding the benefit latency is critical to your decision. You might need to convince others that a long-term return on investment is acceptable or desirable.

Dangerous terrain: Are there other barriers?

Your project team's attitude and personality are among the biggest things that affect the use of practices. Will they accept and adopt the practice or just go through the motions? As with any change, corporate culture also plays a part. Will your management buy in or fight you every step of the way? How will it affect the organizational infrastructure? Identifying what obstacles might lie ahead is as important as clarifying the need for the practice in the first place.

Overconfidence: Can you really implement it?

Finally, there's the bottom line: Do you have the resources and sufficient authority to implement the practice? You can probably imple-

ment something that affects the way your team works internally, but could you implement something like Integrated Product and Process Development with all of its significant effects on other stakeholders? Is there sufficient time left in the project to achieve any benefit? Are there clear instructions on how to implement the practice? Can you find tools or consultants or classes, or will you have to make it up as you go? Does it require convincing three levels of management that it is important? If so, add some time to the benefit latency numbers. You must honestly assess the implementation requirements and your ability to meet them, or you could find yourself unable to complete them and possibly be worse off than you were before.

Even though these questions can be extremely difficult to answer, don't be dissuaded from the hunt. I believe projects and organizations need to keep an ear to the ground and constantly evaluate promising practices. There are good practices out there and we should all try to leverage others' experiences.

Unfortunately, we live in a world where one-size-fits-all and instant gratification is valued. Remember that there really are no silver bullets; neither, as Barry Boehm tells us, are there many lead ones. Barry and I use risk as a fundamental decision factor in selecting between agile and disciplined practices in our upcoming book, *Balancing Agility and Discipline: A Guide for the Perplexed* (Addison Wesley, to be published in 2003).

Keep in mind that every practice has associated cost and benefit, maturity and pedigree, preferred environment, benefit latency, organizational barriers, and required competencies for successful implementation. Make sure the practices you choose fit your needs and your means. Then you've a far better chance of reaping rich rewards from your own best practice treasure hunt.

Richard Turner is a research professor in the Engineering Management and Systems Engineering Department at George Washington University. He serves as assistant deputy director for software engineering and acquisition in the Software Intensive Systems Office of the US Undersecretary of Defense for Acquisitions, Technology, and Logistics. Contact him at Rich.Turner@osd.mil.

CALL FOR PAPERS

Call for Papers: Model-Driven Development

Submission deadline: 15 March
Special Issue Publication: September/October 2003

Model-driven development is the notion that systems can be developed by constructing abstract views of systems and by transforming the resulting models, either automatically or manually, into code. This approach can confer increased productivity and reduced time-to-market by enabling work at a higher level of abstraction and improving communication with non-software domain experts. Common modeling languages and model interoperability herald the replacement of today's code-driven development environments with those based on models.

IEEE Software solicits short articles, written in a straightforward, explanatory style, describing and defining technologies that enable model-driven development, such as

- Model semantics and execution
- Domain-specific language concepts
- Aspect-oriented modeling
- Reverse-engineering
- Model-transformation technologies
- Impediments to adoption
- Combining models with code
- Systems engineering
- Tools, factories, and meta-technologies
- Case studies

Articles must be under 4000 words, including 200 words for each figure and table. A unified, common vocabulary will be encouraged and enforced. We discourage descriptions of specific modeling notations and will refer calls for additions to UML to OMG.

Detailed guidelines are available at http://computer.org/software/edcal.htm#model and from the Guest Editors:

Stephen J. Mellor, Project Technology, USA, steve@projtech.com
Tony Clark, King's College London, UK, anclark@dcs.kcl.ac.uk
Takao Futagami, Toyo Corp., Japan, takao@toyo.co.jp

To submit a manuscript for peer-reviewed consideration, please access the IEEE Computer Society Web-based system, Manuscript Central, at http://cs-ieee.manuscriptcentral.com/index.html. Be sure to select the right manuscript type (Special Issue on Model-Driven Development) when submitting your manuscript. Send a separate email listing the title of your paper and a 75-150 word abstract, but excluding the authors' names. If you wish to be a reviewer, email us at steve@projtech.com, and you will be sent a number of abstracts from which you can choose.